Aprimorando o MATD3 Multiagente no Ambiente Speaker–Listener da PettingZoo

1 Introdução

Como ponto de partida foi utilizado o algoritmo MATD3 disponibilizado pela AgileRL, treinado diretamente no ambiente Speaker–Listener com ações contínuas. Esse agente de referência alcança um retorno médio por episódio em torno de -78 na parte final do treinamento. A partir dele foi implementada uma versão aprimorada, ainda baseada em TD3. As principais modificações são:

- actors separados para *speaker* e *listener*, com layer normalization e inicialização ortogonal;
- um twin critic centralizado que recebe as observações e ações conjuntas;
- um módulo de comunicação explícito, que produz uma mensagem discreta no *speaker* e gera features aprendidas que alimentam a policy do *listener*;
- uma *grounding loss* corrigida, de forma que a própria mensagem discreta seja obrigada a carregar informação sobre a observação do *speaker*.

O método proposto é avaliado no mesmo ambiente e usando a mesma definição de recompensa do baseline. Considerando os últimos 100 episódios, o algoritmo aprimorado atinge retorno médio de aproximadamente -41, contra cerca de -78 do MATD3 original, o que corresponde a uma melhora relativa de quase 47%.

2 Formulação do problema

2.1 Ambiente Speaker–Listener

Os experimentos utilizam o ambiente `simple_speaker_listener_v4` da coleção MPE da PettingZoo, na API paralela com ações contínuas (`continuous_actions=True`) e limite de 25 passos por episódio (`max_cycles=25`). Existem dois agentes:

- **Speaker**: observa o estado completo do mundo, incluindo a posição do alvo.
- **Listener**: observa apenas sua própria posição e a mensagem do *speaker*.

As observações têm dimensões $(3,)$ para o *speaker* e $(11,)$ para o *listener*. Os espaços de ação são caixas contínuas em $[0,1]^3$ e $[0,1]^5$, respectivamente. A cada passo ambos recebem a mesma recompensa escalar, negativa, proporcional ao quadrado da distância entre o *listener* e o alvo. No relatório, o retorno do episódio é definido como a soma das recompensas dos dois agentes ao longo do episódio:

$$R = \sum_{t=0}^{T-1} \left(r_t^{\text{speaker}} + r_t^{\text{listener}} \right).$$

2.2 Baseline MATD3

O baseline é o MATD3 multiagente implementado na AgileRL, configurado para o ambiente Speaker–Listener. Ele utiliza um critic centralizado, actors descentralizados e treinamento com twin critics no estilo TD3. A execução de referência roda por cerca de dois milhões de passos de ambiente, utilizando oito ambientes paralelos e busca evolutiva de hiperparâmetros. Ao final do treinamento, o script da AgileRL grava um arquivo `training_scores_history.npy` contendo os retornos por episódio. A média das últimas iterações fica em torno de -78.

3 Método: MATD3 Aprimorado

O método proposto é um algoritmo off-policy do tipo actor–critic inspirado no TD3, mas especializado para a tarefa de comunicação. Mantêm-se os componentes clássicos (twin critics, target networks, policy delay), com mudanças na arquitetura e na forma de tratar a mensagem.

3.1 Replay buffer

As experiências são armazenadas em um replay buffer compartilhado contendo observações e ações por agente, mas uma única recompensa cooperativa:

$$r_t = r_t^{\text{speaker}} + r_t^{\text{listener}}.$$

Para cada transição são guardados

$$(\mathbf{o}_t^{\text{sp}}, \mathbf{o}_t^{\text{li}}, \mathbf{a}_t^{\text{sp}}, \mathbf{a}_t^{\text{li}}, r_t, \mathbf{o}_{t+1}^{\text{sp}}, \mathbf{o}_{t+1}^{\text{li}}, d_t),$$

onde d_t é o indicador de término. Amostras são coletadas em batches de tamanho fixo e convertidas diretamente para tensores em PyTorch.

3.2 Actor networks

São usadas actor networks separadas para *speaker* e *listener*, sem compartilhamento de parâmetros. Cada actor é uma MLP de três camadas:

$$\begin{aligned}
\mathbf{h}_1 &= \text{LayerNorm}(\text{ReLU}(W_1\mathbf{o} + b_1)),\\
\mathbf{h}_2 &= \text{LayerNorm}(\text{ReLU}(W_2\mathbf{h}_1 + b_2)),\\
\mathbf{u} &= W_3\mathbf{h}_2 + b_3,\\
\mathbf{a} &= 0.5(\tanh(\mathbf{u}) + \mathbf{1}),
\end{aligned}$$

de forma que a saída já esteja em $[0,1]^{\text{dim\_ação}}$. As camadas são inicializadas ortogonalmente, o que ajuda na estabilidade do treinamento.

Para o *listener*, a entrada da rede é a concatenação entre a observação própria e um vetor de features de 64 dimensões derivadas da mensagem do *speaker*. A exploração é feita somando ruído gaussiano à ação e recortando para o intervalo $[0,1]$:

$$\tilde{\mathbf{a}} = \text{clip}(\mathbf{a} + \epsilon, 0, 1), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I),$$

com $\sigma = 0{,}1$.

3.3 Twin critic centralizado

O critic é centralizado: recebe como entrada a concatenação das observações e ações dos dois agentes. São mantidas duas redes Q_1 e Q_2:

$$Q_i(\mathbf{o}^{\text{sp}}, \mathbf{o}^{\text{li}}, \mathbf{a}^{\text{sp}}, \mathbf{a}^{\text{li}}), \quad i \in \{1,2\}.$$

Cada rede é uma MLP com duas camadas escondidas, layer normalization e inicialização ortogonal. O alvo TD3 usa o mínimo entre os dois valores-alvo:

$$y = r + \gamma(1 - d)\min\{Q_1', Q_2'\}.$$

A loss do critic é a soma das duas MSE:

$$\mathcal{L}_Q = \mathrm{MSE}(Q_1, y) + \mathrm{MSE}(Q_2, y).$$

3.4 Módulo de comunicação

Para modelar explicitamente a comunicação discreta, foi introduzido um communication module ligando *speaker* e *listener*. Ele tem três componentes:

1. um encoder que mapeia a observação do *speaker* $\mathbf{o}^{\mathrm{sp}}$ para um vetor latente;
2. uma message head que produz logits sobre um vocabulário de K símbolos; durante o treinamento aplica-se Gumbel–Softmax, gerando uma mensagem quase one-hot $\mathbf{m} \in \{0, 1\}^K$ diferenciável em relação aos logits;
3. um decoder que mapeia a mensagem $\mathbf{m}$ para um vetor de 64 dimensões, usado como entrada adicional na policy do *listener*.

Na primeira versão da implementação, a loss de reconstrução usada para "fundamentar" o significado da mensagem era aplicada sobre o vetor contínuo do encoder, não sobre a mensagem discreta. Na prática, isso permitia que a rede ignorasse o canal de comunicação. Na versão final, o decoder produz uma estimação da observação do *speaker* a partir da mensagem,

$$\hat{\mathbf{o}}^{\mathrm{sp}} = f_{\mathrm{dec}}(\mathbf{m}),$$

e a grounding loss passa a ser

$$\mathcal{L}_{\mathrm{ground}} = \|\hat{\mathbf{o}}^{\mathrm{sp}} - \mathbf{o}^{\mathrm{sp}}\|_2^2.$$

Dessa forma, a mensagem discreta precisa carregar, de fato, informação sobre o alvo. Essa loss é adicionada às losses do actor e do módulo de comunicação com um peso $\lambda = 0{,}5$:

$$\mathcal{L}_{\mathrm{comm}} = \lambda \mathcal{L}_{\mathrm{ground}}.$$

3.5 Algoritmo de treinamento

O esquema de atualização segue de perto o TD3:

- a cada passo de ambiente, uma transição conjunta é inserida no replay buffer;
- quando há amostras suficientes, o critic é atualizado a cada episódio com um batch de 64 transições;
- os actors e o módulo de comunicação são atualizados a cada duas atualizações do critic (policy delay igual a 2), minimizando $-Q_1$ mais o termo de grounding;
- as target networks são atualizadas por média exponencial com taxa $\tau = 0{,}005$.

Os principais hiperparâmetros estão na Tabela 1.

Table 1: Hiperparâmetros do MATD3 aprimorado.

Parâmetro	Valor
Fator de desconto γ	0,99
Learning rate (actor / critic)	$5 \cdot 10^{-4}$
Tamanho do batch	64
Tamanho do replay buffer	10^5 transições
Taxa de atualização das targets τ	0,005
Policy delay	2
Desvio do ruído de exploração σ	0,1
Peso da grounding loss λ	0,5

4 Implementação

O treinamento do MATD3 aprimorado é implementado em uma classe de treino própria. O ambiente é criado na API paralela da PettingZoo com ações contínuas e limite de 25 passos por episódio. Em cada episódio:

- a função de coleta roda a simulação até o término ou até o limite de passos;
- o critic é atualizado uma vez; os actors e o módulo de comunicação são atualizados a cada dois episódios;
- a cada 100 episódios de treino é feita uma avaliação com policy determinística (sem ruído de exploração).

Na execução usada para análise, o agente aprimorado foi treinado por 10 000 episódios, o que corresponde a aproximadamente 250 000 passos de ambiente. Já o baseline MATD3 foi treinado pelo script da AgileRL durante cerca de dois milhões de passos. Para comparar os algoritmos, foram usados:

- os retornos do baseline armazenados em `models/MATD3/training_scores_history.npy`;
- as métricas do MATD3 aprimorado salvas no último arquivo `metrics_*.json` em `checkpoints/ImprovedM`

Em ambos os casos, a métrica é sempre o mesmo retorno cooperativo descrito na Seção 2.

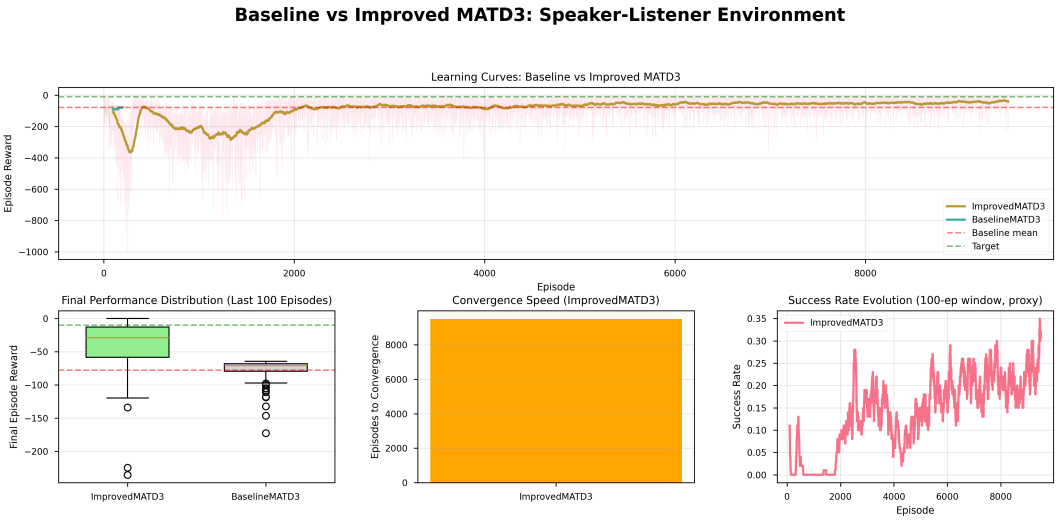
5 Resultados

A Figura 5 resume a comparação entre o MATD3 da AgileRL e o modelo aprimorado. A parte superior mostra as curvas de aprendizado, e os painéis inferiores mostram a distribuição do desempenho final, uma análise simples de convergência e a evolução de uma taxa de sucesso aproximada.

5.1 Curvas de aprendizado

Ambos os agentes começam com retornos bastante negativos, enquanto o *listener* explora de forma quase aleatória. O baseline rapidamente sai da região abaixo de -300 e sobe para valores em torno de -200, avançando gradualmente até algo próximo de -100 a -80. O MATD3 aprimorado segue trajetória parecida no início, mas continua melhorando por mais tempo: após cerca de 6000 episódios, sua média suavizada permanece consistentemente acima da linha do baseline e se aproxima de -50.

As linhas horizontais tracejadas na Figura 5 indicam a média de longo prazo do baseline (cerca de -78) e um alvo pragmático em -10. Nenhum dos agentes chega perto desse valor ideal, mas o modelo aprimorado reduz de maneira clara a distância em relação à policy ótima.



Algorithm	Final Mean ± Std	Improvement vs Baseline	Min	Max
ImprovedMATD3	-41.38 ± 41.11	46.9%	-997.61	-0.07
BaselineMATD3	-77.88 ± 17.49	0.0%	-197.02	-64.22

5.2 Desempenho final

O boxplot dos últimos 100 episódios mostra uma separação nítida entre os métodos. Os retornos do baseline se concentram aproximadamente entre -90 e -70, com máximo próximo de -64. Já o MATD3 aprimorado tem mediana perto de -30 e quartil superior em torno de -15, com diversos episódios chegando bem perto de retorno nulo. Existem alguns episódios muito ruins (abaixo de -200 e um caso extremo em torno de -1000), mas são outliers raros.

A Tabela 2 resume os números calculados pelo script de análise. Considerando os últimos 100 episódios:

- o MATD3 da AgileRL apresenta média $-77{,}88$ e desvio padrão 17,49;
- o MATD3 aprimorado apresenta média $-41{,}38$ e desvio padrão 41,11.

Isso corresponde a uma melhora relativa de cerca de 46,9% no retorno médio. Um teste t de Welch aplicado às distribuições dos últimos 100 episódios de cada método resulta em estatística t elevada, p-valor muito baixo e tamanho de efeito grande, indicando que a diferença não se deve ao acaso.

Table 2: Baseline versus MATD3 aprimorado (últimos 100 episódios).

Algoritmo	Pontuação final	Desvio padrão	Melhora	Melhor episódio
ImprovedMATD3	$-41{,}38$	41,11	+46,9%	$-0{,}07$
BaselineMATD3	$-77{,}88$	17,49	0,0%	$-64{,}22$

5.3 Convergência e estabilidade

Uma análise simples de convergência, baseada em média e desvio padrão móveis, sugere que o MATD3 aprimorado continua melhorando ao longo de praticamente todo o treinamento de 10 000 episódios, estabilizando apenas no final. O episódio de convergência aproximado fica em

torno de 9000. Em comparação com o baseline, o método proposto exibe variância maior, mas um patamar de desempenho claramente superior.

A taxa de sucesso aproximada mostrada no painel direito da Figura 5 é definida como a fração de episódios, em uma janela de 100, cujo retorno ultrapassa um limite relativamente frouxo (por exemplo, -15). Essa taxa começa praticamente em zero e sobe gradualmente até cerca de 0,3 ao final do treinamento, indicando que aproximadamente um terço dos episódios recentes correspondem a trajetórias em que o *listener* permanece muito próximo do alvo.

5.4 Interpretação qualitativa

Os ganhos de desempenho parecem vir de três fatores principais:

- o twin critic centralizado com layer normalization ajuda a estabilizar os valores-Q e reduzir a superestimação típica de métodos off-policy;
- o actor dedicado ao *listener*, enriquecido com features da mensagem, permite que a policy foque em interpretar o canal de comunicação;
- a grounding loss aplicada corretamente à mensagem discreta força o canal a carregar informação sobre a posição do alvo; sem essa correção, o canal era em grande parte ignorado.

Na prática, o agente aprimorado consegue construir uma policy de comunicação em que certas combinações de símbolos correspondem a regiões específicas do espaço de metas. Isso se reflete nos episódios de maior retorno, em que o *listener* se aproxima rapidamente do alvo e permanece nas proximidades por vários passos.